Informe práctica 1

Servidor y cliente HTTP/HTTPS

Álvaro Navarro Martínez

29/10/2023

Índice

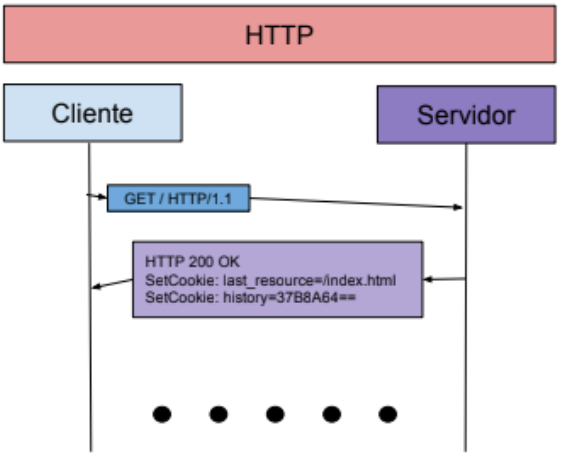
Introducción

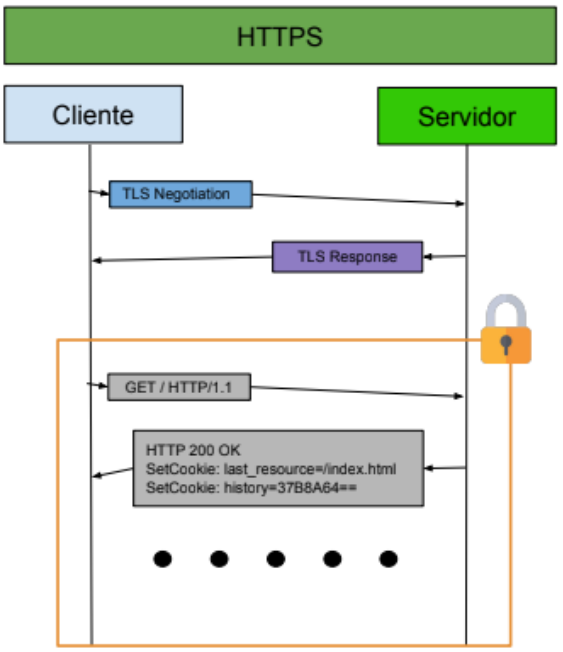
En este trabajo, se ha llevado a cabo una implementación integral de un sistema de comunicación cliente-servidor en Java, que abarca tanto el protocolo HTTP como su versión segura, HTTPS. Además, se ha incorporado un servicio de historial de accesos basado en cookies para mantener un seguimiento de las interacciones. La arquitectura multihilo garantiza una eficiencia óptima, mientras que la creación de certificados asegura la confidencialidad y autenticación en las comunicaciones. El enfoque meticuloso en la implementación de cabeceras y cuerpo de HTTP demuestra un compromiso con la precisión y la calidad en el código desarrollado.

Funcionalidades

Obligatorias

Comunicación





Concurrencia

Se ha implementado **concurrencia por parte del servidor**, para que **cada** vez que reciba una **conexión** de un cliente distinto, se **cree un hilo que procese sus peticiones**. Esto consigue que podamos tener **muchos clientes conectados** de manera simultánea.

Operaciones de entrada/salida

De entre las distintas opciones para la implementación de las operaciones de entrada y salida, se ha escogido la opción de la entrada/salida **no bloqueante**. Esto permite que un cliente que no aproveche su tiempo de escritura en el buffer, pasado un tiempo, se cierre la conexión. De esta manera, el servidor consigue mejorar su rendimiento, porque no tendrá clientes ociosos

malgastando sus recursos y ahorra recursos en todo momento, solo se utilizan si son necesarios.

Control de errores

Aunque no era necesario, se ha implementado la funcionalidad necesaria para tener un pequeño **control de errores** por peticiones incorrectas, como son la entrada de **versiones no soportadas**, **peticiones mal formateadas** y control de **métodos permitidos**.

Procesamiento de las cookies

Cuando llega una petición de un cliente, se extraen las **cookies** incluidas en él para su **procesamiento**. En el **caso de no incluirse**, se añaden las cookies de

- last_reource=<recurso_solicitado>
- historial=<recurso_solicitado>=1

En el **caso de incluirse cookies** en la petición, se van a procesar de la siguiente manera

```
Para cada cabecera de Cookie del cliente incluida
do
    si "last_resource" entonces encontrado_lr = true
    si "history" entonces procesarHistorial()
done
```

Este procesamiento de historial se hace convirtiendo la cadena incluida en la cabecera en un **Mapa<String, Integer>** de Java, de manera que se pueden almacenar tanto los **recursos** como las **veces solicitadas**, y una vez que el mapa está creado, vamos a **añadir el último recurso a él**, o bien añadiendo una nueva entrada porque el recurso aún no existía, o bien añadiendo un acceso a un recurso ya solicitado anteriormente.
Esta cadena que se manda en la cookie está **formateada en Base64**, para que el cliente solo tenga que almacenarla y no procesarla en ningún momento.

Visión por pantalla

El funcionamiento se ha hecho para que el Mapa creado con las cookies pueda ser **interpretado por un creador de ficheros HTML** y convertir el mapa en una **tabla**. De esta manera conseguimos el resultado siguiente:

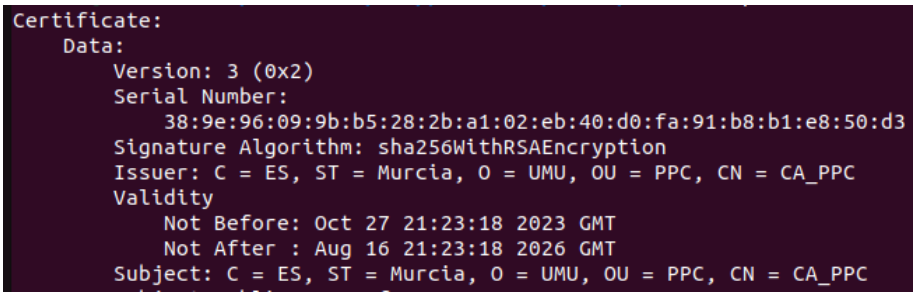
Fichero solicitado: /paisaje.txt

Historial:

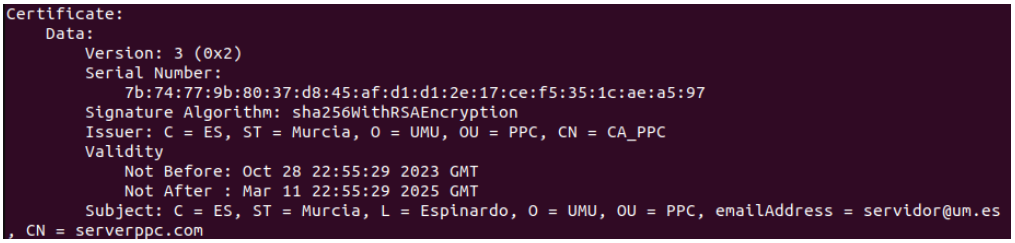
Recurso	Numero de peticiones
/favicon.ico	2
/hola.html	8
/index.html	18
/paisaje.jpg	5
/paisaje.txt	4

Generación de certificados

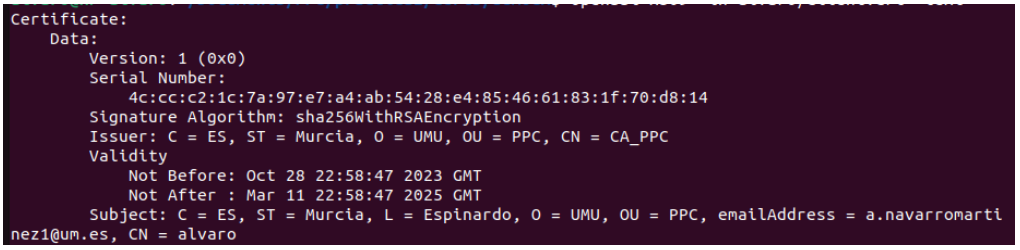
Los certificados han sido creados usando una autoridad de certificación **CA_PPC**.



Con ella se han **emitido** los **certificados del servidor** (serverppc.crt)



Y el **certificado del cliente** (alvaro.crt).



Una vez se han tenido estos certificados, hemos convertido estos en **formato PKCS12**, y así se han importado a los navegadores y al servidor.

Opcionales

Contador de número de acceso a recurso

Como se ha explicado en el procesamiento de las cookies, se implementa un Mapa para poder contar el número de peticiones a un determinado recurso

Persistencia de cookies en el cliente

Para que entre distintas conexiones el cliente pueda tener las mismas cookies, manteniendo su historial, se ha **desarrollado un gestor de persistencia de las cookies** en un fichero con el siguiente aspecto:

```
1 history=JTdCJTJGaW5kZXguaHRtbCUzRDIlMkMlMkZwcnVlYmElM0QzJTdE; max-age=3600,2023-10-29 10:59:08
2 last_resource=/prueba; max-age=3600,2023-10-29 10:59:08
```

Este fichero almacena las cookies que son asignadas por el servidor, y el **gestor de cookies** se encarga de cargarlas en las peticiones HTTP **si siguen siendo válidas**. Una vez se cumpla el **max-age**, no se cargan en las peticiones y se eliminan del fichero.

Persistencia de conexiones del servidor

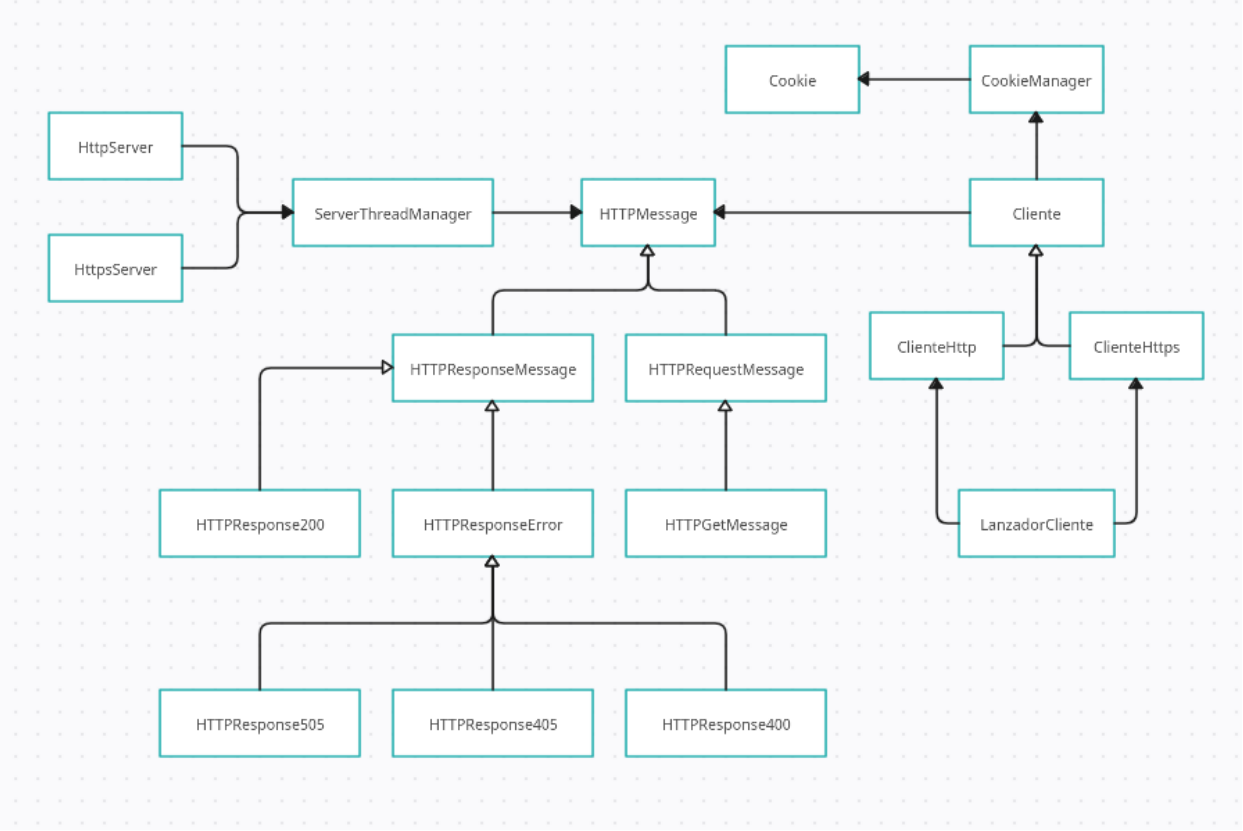
Para ser más eficientes, en lugar de abrir una conexión por cada recurso solicitado; se **abrirá una conexión en la que se pueden pedir varios recursos**, esto se consigue almacenando el estado de las peticiones en cada momento y controlandolas, de la siguiente manera

```
inicializarConexión()
while conexion abierta
do
    recibirMensaje()
    procesarMensaje()
    enviarRespuesta()
done
cerrarConexión()
```

Timeout

Para que la entrada y salida no sea bloqueante, se implementa un mecanismo de timeout detallado en el apartado entrada/salida

Implementación



Ejecución

Para poder ejecutar el cliente y servidor es necesario **arrancar ambos servidores** en primer lugar, y después **arrancar el cliente**. Una vez que el cliente está lanzado, nos deja **elegir el método con el que se establece conexión con el servidor**: HTTP y HTTPS; elegimos el método y podemos empezar a hacer peticiones.
Para que se pueda ver de manera **más visual**, podremos ejecutar un cliente como el navegador **firefox**, que interpreta el código HTML enviado.